Project Proposal for Distributed Systems: RSA Codebreaker

Student Mark Porter

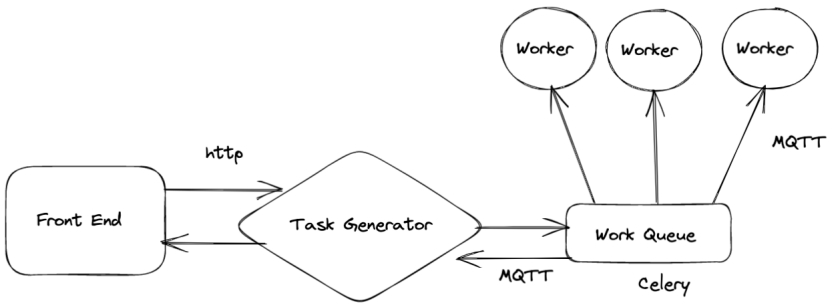
Problem Description

Cryptography is used in many different applications to protect private information. The problem we choose to tackle is to decipher a simple RSA encrypted message that is provided by the user without having access to its private key. In this way we will attempt to "break" very weak RSA encryption. An example of the type of problems we are looking to solve can be found at https://asecuritysite.com/rsa/rsac.

This will be a toy example which can be used to decrypt messages from user input. One object will be to make the solution scale in a way that makes this decryption happen faster depending on how many resources are provided. The first challenge of a project like this will be to generate a list of primes which can be used factor the RSA encrypted public key. Another problem will be to do the checking to see if these numbers are factors of the given encrypted message. This problem is well understood but implementing it in a distributed system will provide good opportunity to explore and test and performance impacts of different design decisions.

Proposed Solution

We propose a solution where we will have a very simple front end for user input, which will constitute one node. This will communicate with a task generator node over HTTP which will be responsible for creating factorization tasks and also for returning the result. We will have a work queue that uses MQTT to communicate new jobs to a series of worker nodes. This can be seen in the diagram below. In this way we will implement a distributed algorithm for cracking RSA codes.



Explanation of components

Front End: This component will be a specialized GUI written in HTML that allows the user to submit new RSA challenges to the system. It will be quite simple and basically just expose the HTTP API of the task generator.
Task Generator: This component will be responsible for turning the RSA code into a series of broken down factorization tasks that can then be used to determine the prime factors of the number, thus allowing us to break the RSA code.
Worker Queue: This will be a Celery worker queue that uses RabbitMQTT as a backend / broker. It will allow us to dole out tasks to our worker nodes and also return the result of the culmination of their work.
Worker: The workers will be responsible for very simple factorization tasks. They will determine whether a number is prime. This will allow us to use a distributed algorithm for prime factorization.

Scalability

We aim for our proposed solution to be highly scalable. By allowing basically unlimited increase of the number of worker nodes. We hope that our distributed algorithm can be implemented in such a way that any increase in load

(through increase of the size of the input) will be able to be matched by scaling of the number of workers. This will not be done on a dynamic basis as in some systems as that would be out of scope for this project. Instead, we will allow static configuration which determines the number of available worker nodes for the system.

Metrics

We will evaluate our system based on several metrics. The first of which being how quickly it can crack a simple RSA Code. We will also attempt to establish whether adding extra worker nodes makes the system faster. Another evaluation we will perform is how large of a number the code breaker can crack in a reasonable timeframe. It will be interesting to see if, as we scale the number of worker nodes, we can crack larger codes.